Angewandte Mathematik und Informatik

Universität zu Köln

Report No. 00.395

A Paint Shop Problem for Words

by

Th. Epping, W. Hochstättler, P. Oertel

Center for Parallel Computing
Universität zu Köln
D–50923 Köln
GERMANY
{epping,wh,oertel}@zpr.uni-koeln.de

1991 Mathematics Subject Classification: 90B30, 90C39
Keywords: Assembly line, Dynamic programming, NP-completeness, Paint shop, Sequencing

A Paint Shop Problem for Words

Th. Epping, W. Hochstättler, P. Oertel*

August 23, 2000

Abstract

Motivated by an application in the automobile industry, we present results on the following problem:

Given a word $w = (w_1, \ldots, w_n)$ and a color vector $f = (f_1, \ldots, f_n)$ with f_i denoting the color of w_i, find a permutation $\sigma \in \mathcal{S}_n$ such that $w_{\sigma(i)} = w_i$ and the number of color changes within $\sigma(f) = (f_{\sigma(1)}, \ldots, f_{\sigma(n)})$ is minimized.

We show that this problem is polynomially solvable if we bound both, the number of letters and the number of colors, and becomes $\mathcal{NP}$-complete otherwise.

1 Introduction

In Europe, customers demand individually furnished cars, therefore cars are generally built on order. The sequencing of those orders has great impact on production quality and cost. We focus on a subproblem of this process: the painting of the car bodies.

Due to numerous production requirements the sequence, in which the bodies enter the paint shop, must be considered an external parameter. Our aim is to enamel these bodies, such that the scheduled orders can be met. For each color change in the sequence the color jets have to be cleaned, giving rise to non negligible cost and pollution. Therefore, the minimization of color changes is aspired by the automobile industry for a long time (see [2] and references therein). So far, mainly heuristics which make extensive use

*supported by Alfried Krupp von Bohlen und Halbach Stiftung

of interim storages are used to deal with this complex task. We drop the usage of color storages and analyze the following basic mathematical model of the problem. We associate with each body style a letter of an alphabet and interpret a word w as a sequence of car bodies. A vector f of the same length as w determines a sequence of car colors.

Definition 1.1 *Given a set F of colors and a vector $f = (f_1, \ldots, f_n)$ with $f_i \in F$ for $i = 1, \ldots, n$, we define*

$$b_j := \begin{cases} 0\,, \text{ if } f_j = f_{j+1} \\ 1\,, \text{ otherwise } (j = 1, \ldots, n-1) \end{cases}$$

Then we denote by $\gamma(f) := \sum_{j=1}^{n-1} b_j$ the number of color changes within f.

Now, the problem is to find a permutation of the color sequence, such that the number of color changes is minimized.

Problem 1.1 *A Paint Shop Problem for Words (PPW)*

Instance *A finite alphabet Σ, a word $w = (w_1, \ldots, w_n) \in \Sigma^*$, a set F of colors and a coloring $f = (f_1, \ldots, f_n)$ of w.*

Question *Find a permutation $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that $w_{\sigma(i)} = w_i$ for $i = 1, \ldots, n$ and $\gamma(\sigma(f))$ is minimized.*

Note, that the initial coloring of a word serves only to determine the reservoir of letters available in each color. Thus, when constructing an instance of PPW, we need only specify these reservoirs instead of giving an explicit color vector.

In the following we first present two complexity results on PPW and then give an $\mathcal{O}(|F| n^{(|F|-1)|\Sigma|})$ dynamic program for its solution. Throughout the paper we use standard notation as introduced in [1].

2 Complexity Results

Theorem 2.1 *PPW is $\mathcal{NP}$-complete for $|F| = 2$.*

Proof We give a reduction from 3SAT [1]. Let $C = \{c_1, \ldots, c_m\}$ denote the set of clauses and $V = \{v_1, \ldots, v_n\}$ denote the set of variables of an instance of 3SAT. We construct an instance of PPW as follows. The word w consists of n *variable blocks*. We explain the construction of a single variable block for a fixed variable v_i in detail.

We assume that v_i appears in clauses $c_{i_1}, \ldots, c_{i_k}$ as literals $l_{i_1}, \ldots, l_{i_k}$ and wlog. literals are sorted, such that $l_{i_j} = v_i$ for $j = 1, \ldots, r$ and $l_{i_j} = \bar{v}_i$ for $j = r+1, \ldots, k$. Note, that we can always assume $r \geq 1$ and sort the literals independently for each variable block. The variable block of v_i then consists of k *literal blocks* $b_{i_1}, \ldots, b_{i_k}$. The letters we use for building the variable block are given as follows.

- First, we provide $4k$ *variable letters* L_j^i $(j = 1, \ldots, 2k)$, each once in color red and once in color blue. Variable letters differ for each variable block. In the following, we drop their superscript whenever it is clear which variable we refer to.
- Second, we introduce $3m$ *satisfaction testers* T_s $(s = 1, \ldots, m)$, each once in color red and twice in color blue. Each satisfaction tester T_s is associated with a clause c_s. We use these satisfaction testers T_{i_j} $(j = 1, \ldots, k)$, one of each, within the variable block.
- Finally, we introduce *separators* Z, each available only in blue. Separators are the same in all variable blocks. We use $k+1$ of them within the variable block.

We arrange the letters of the variable block as follows. For each literal block we provide four variable letters, as indicated in Figure 1. We precede the first variable letter of a literal block b_{i_j} with ZT_{i_j} for $j = 1, \ldots, r$ and precede the third variable letter of a literal block b_{i_j} with ZT_{i_j} for $j = r+1, \ldots, k$. Finally, we add a separator Z behind the last variable letter of a variable block.

As each variable letter L_j is available only once in each color, it is guaranteed that the two variable letters L_{2j} in each literal block b_{i_j} will have different colors. We claim that we can get by with two color changes for every literal block and prove the following:

There exists a satisfying truth assignment for $v_1, \ldots, v_n$ if and only if there exists a coloring of w with exactly $6m$ color changes.

Given a satisfying truth assignment, let $c_s = \{l_{s_1}, l_{s_2}, l_{s_3}\}$ be a clause and assume that l_{s_1} satisfies the clause. We then color the associated satisfaction

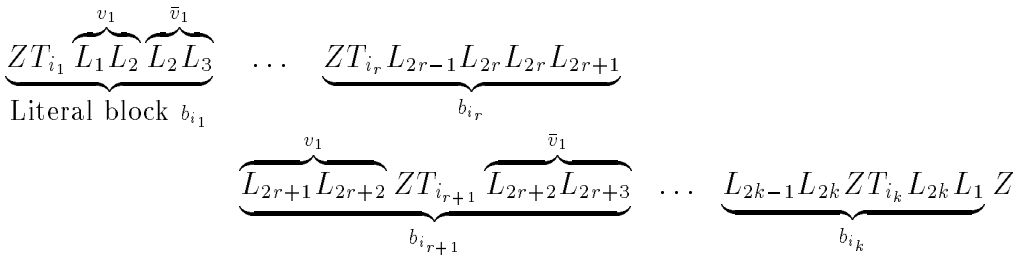

Figure 1: The structure of a variable block.

tester T_s in the literal block of l_{s_1} red (and have to color the satisfaction testers in the literal blocks of l_{s_2} and l_{s_3} blue). Additionally, we color the variable letters of the variable block alternatingly (red, red, blue, blue, ...), if the variable related to the variable block is set to TRUE in the truth assignment; if the variable is set to FALSE, we use the alternating color scheme (blue, blue, red, red, ...).

If our coloring starts with two blue variable letters and we come across a color change between two literal blocks, we always account it to the number of color changes within the preceeding one (including the color change between the last variable letter and the final separator). Otherwise, if our coloring starts with two red variable letters and we come across a color change between two literal blocks, we always account it to the number of color changes within the succeeding one. In both cases, this approach results in exactly two color changes within any literal block. Thus, we end up with $6m$ color changes altogether for a coloring of w.

Now suppose that we are given a coloring with exactly $6m$ color changes. First note, that counting in a similar way like above we have at least two color changes per literal block. As we know that we have a total of $3m$ literals, we have exactly two color changes per literal block. This if only possible if the variable letters within each variable block are colored in connected blocks of size two, resulting in a coloring sequence of either (red, red, blue, blue, ...) or (blue, blue, red, red, ...).

We then assign the value TRUE to variables whose variable block starts with two red variable letters, and FALSE otherwise and show that we get a satisfying truth assignment. Let $c_s = \{l_{s_1}, l_{s_2}, l_{s_3}\}$ be a clause and assume that the associated satisfaction tester T_s in the literal block of l_{s_1} has been colored red. In order not to give rise to more than two color changes in this literal block, the satisfaction tester must be followed by a red letter. This implies

that l_{s_1} is positive if and only if its variable has been set to TRUE. Thus, the formula is satisfied. □

If we bound the size of the alphabet instead of the number of colors, we get a similar result.

Theorem 2.2 *PPW is $\mathcal{NP}$-complete for $|\Sigma| = 2$.*

Proof We give a pseudo-polynomial reduction from 3-PARTITION (see [1], Lemma 4.1, pp. 101). Let $A = \{a_1, \ldots, a_{3m}\}$ denote the set of elements with size $s(a_1), \ldots, s(a_{3m}) \in \mathbb{Z}^+$ and $B \in \mathbb{Z}^+$ denote the bound of an instance of 3-PARTITION.

We construct an instance of PPW as follows. For each element a_i of size $s(a_i)$ we provide $s(a_i)$ times the letter L in color $f(a_i)$, where $f(a_i) \neq f(a_j)$ if $i \neq j$. Additionally, we provide $m-1$ times the letter Z in color f_0. The word w then consists of m *partition blocks* $b_1, \ldots, b_m$ of size B that contain the letter L and are separated by the letter Z (see Figure 2).

$$\underbrace{L \ldots L}_{B} Z \underbrace{L \ldots L}_{B} Z \ldots Z \underbrace{L \ldots L}_{B}$$

Figure 2: The general structure of w.

Clearly, there exists a partition of A into m disjoint sets $S_1, \ldots, S_m$ such that $\sum_{a \in S_j} s(a) = B$ for $j = 1, \ldots, m$ if and only if there exists a coloring of w with $4m-2$ color changes. □

3 A Dynamic Program

Given an instance of PPW, we can solve this instance by a dynamic program. We pass through the given word w letter by letter from the left to the right and record each possible coloring up to the current position in a state of our dynamic program. Each state is given as

$$\gamma_a = \gamma_a(l_1^1, \ldots, l_{|F|}^1, l_1^2, \ldots, l_{|F|}^2, \ldots, l_1^{|\Sigma|}, \ldots, l_{|F|}^{|\Sigma|}, k) \text{ with } a = \sum l_j^i,$$

where l_j^i counts the occurences of letter i in color j in the recorded coloring and k denotes the color that has been used to color the letter w_a. The value

$m(\gamma_a)$ of a state γ_a is the minimal number of color changes of the partial word with the recorded colors. Note, that given a state γ_a the letter i that has to be colored next is determined by $i = w_{a+1}$. Finally, we denote by r_j^i the initial reservoir of letter i available in color j for a coloring of w.

We then apply the dynamic program depicted in Figure 3.

1. `Create`
$$\gamma_0(0, \ldots, 0, k)$$
`and initialize`
$$m(\gamma_0(0, \ldots, 0, k)) = 0$$
`for all` $k \in F$.

2. `For` $a \leq n$ `do:`
`If` $(r_f^{w_a} - l_f^{w_a}) > 0$, `create`
$$\gamma_a(\ldots, l_f^{w_a} + 1, \ldots, f)$$
`and set`
$$m(\gamma_a(\ldots, l_f^{w_a} + 1, \ldots, f)) = \min_{k \neq f}\{m(\gamma_{a-1}(\ldots, l_f^{w_a}, \ldots, f)), m(\gamma_{a-1}(\ldots, l_f^{w_a}, \ldots, k)) + 1\}$$
`for all` $f \in F$.

Figure 3: The dynamic program for PPW.

Theorem 3.1 *The dynamic program depicted in Figure 3 solves an instance of PPW for an alphabet Σ and a color set F, requiring a space and time complexity of $\mathcal{O}(|F|n^{|F||\Sigma|})$.*

Proof We proceed by induction on a and show, that $m(\gamma_a)$ is the minimal number of color changes that appear if we use the coloring recorded in γ_a for a coloring of the first a letters of w. Clearly, the initialization $m(\gamma_0) = 0$ for $a = 0$ is correct. If $a > 0$, the letter that has to be colored next for creating a state γ_a is w_a. We can color w_a in color f only if there is at least one letter w_a in color f left. Thus $r(w_a, f) - l_f^{w_a}$ has to be positive. Then $m(\gamma_a)$ is the minimum of $m(\gamma_{a-1})$, where in state γ_{a-1} the letter w_{a-1} has been colored with the same color as w_a, and $m(\gamma_{a-1}) + 1$, where in state γ_{a-1} the letter

w_{a-1} has been colored with a different color as w_a, increased by one due to the additional color change.

As every l_j^i is bounded from above by $l_j^i \leq n$, we directly get $n^{(|F||\Sigma|)}$ as an upper bound for the number of states that can be created. The computation of $m(\gamma_a)$ requires a complexity of $\mathcal{O}(|F|)$. □

Note, that it is sufficient to record l_j^i only for $|F|-1$ colors. Thus we can improve on the complexity of the dynamic program.

Corollary 3.1 *The dynamic program can be implemented to run in* $\mathcal{O}(|F|n^{(|F|-1)|\Sigma|})$.

References

[1] M. R. Garey and D. S. Johnson: *Computers and Intractability. A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[2] S. Spieckermann and S. Voß: *Paint Shop Simulation in the Automotive Industry.* ASIM Mitteilungen 54 (1996), pp. 367-380.